알튜비튜
투 포인터

두 개의 포인터로 배열을 빠르게 탐색하는 알고리즘입니다.
코딩 테스트에선 주로 효율성을 보는 문제에 활용됩니다.

이와 더불어 투 포인터와 함께 자주 활용되는 누적 합, 슬라이딩 윈도우에 대해서도 알아봅니다.

도전 문제 1

#알튜비튜

20437번: 문자열 게임 2 - Gold 5

문제

- 알파벳 소문자로 이루어진 문자열 W와 양의 정수 K가 주어짐
- 어떤 문자를 정확히 K개 포함하는 가장 짧은 연속 문자열의 길이와
  어떤 문자를 정확히 K개 포함하고, 문자열의 첫 번째와 마지막 글자가 해당 문자로 같은
  가장 긴 연속 문자열의 길이를 구하는 문제

제한 사항

- 게임의 개수 T : $1 \le T \le 100$
- 문자열의 길이 |W| : $1 \le |W| \le 10,000$
- 정수 K : $1 \le K \le |W|$

도전 문제 1

#알튜비튜

예제 입력

```
2
superaquatornado
2
abcdefghijklmnopqrstuvwxyz
5
```

예제 출력

```
4 8
-1
```

예제 입력

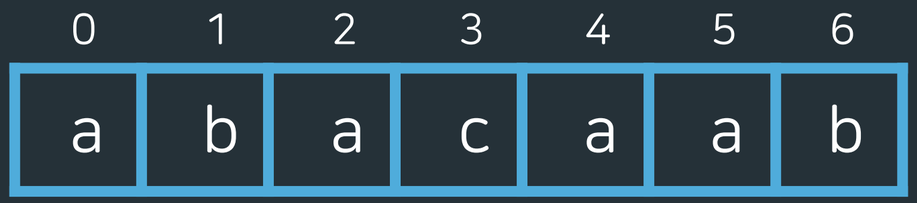
```
1
abaaaba
3
```

예제 출력

```
3 4
```

어떤 문자를 정확히 K개 포함하는 연속 문자열

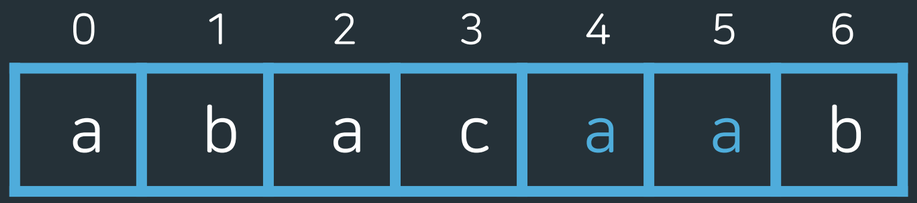
k = 2



어떤 문자를 정확히 K개 포함하는 연속 문자열

#알튜비튜

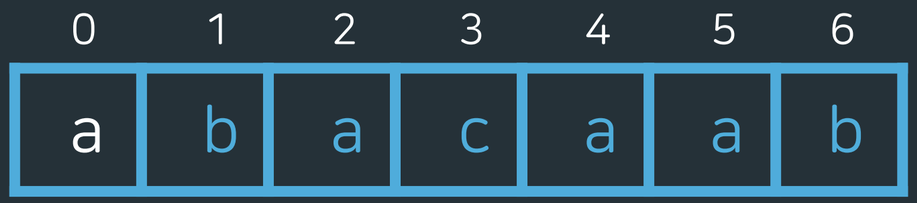
k = 2



가장 짧은 연속 문자열 → aa

어떤 문자를 정확히 K개 포함하는 연속 문자열

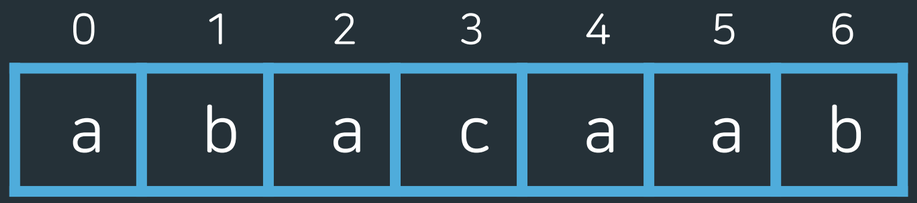
k = 2



가장 긴 연속 문자열 → bacaab

포인터는 어디에…?

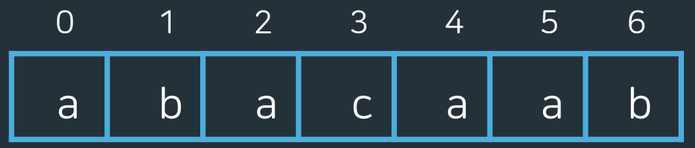
k = 2

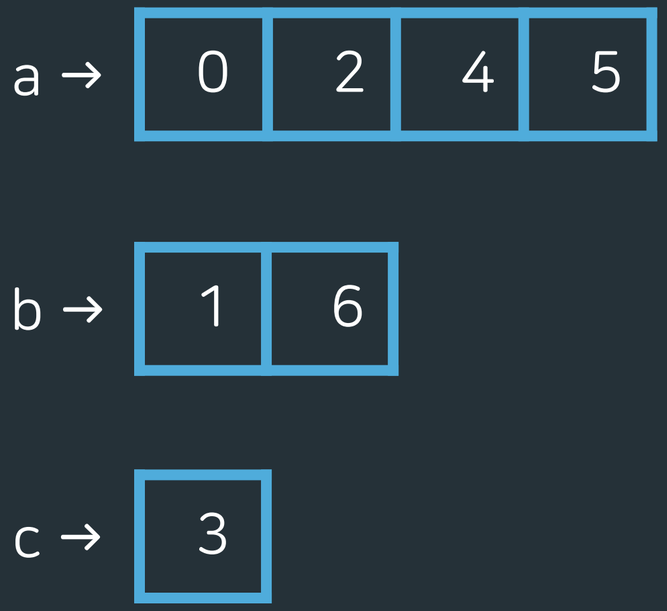


가장 짧은 연속 문자열 → aa
가장 긴 연속 문자열 → bacaab

연속 문자열 내의 해당 문자의 위치만 확인

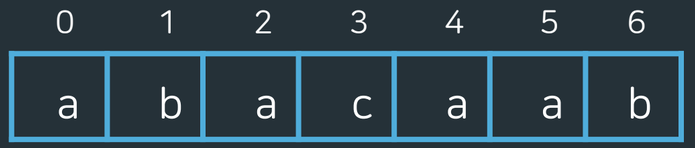
문자별로 나눈다면

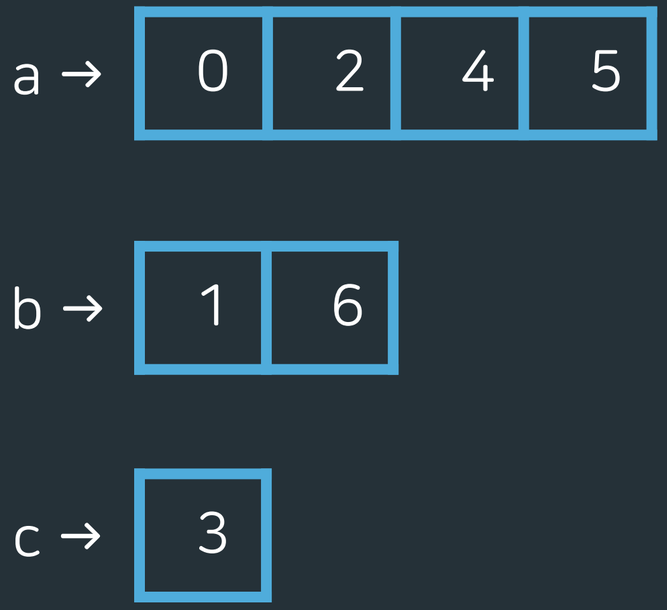




문자별로 등장 위치를 저장

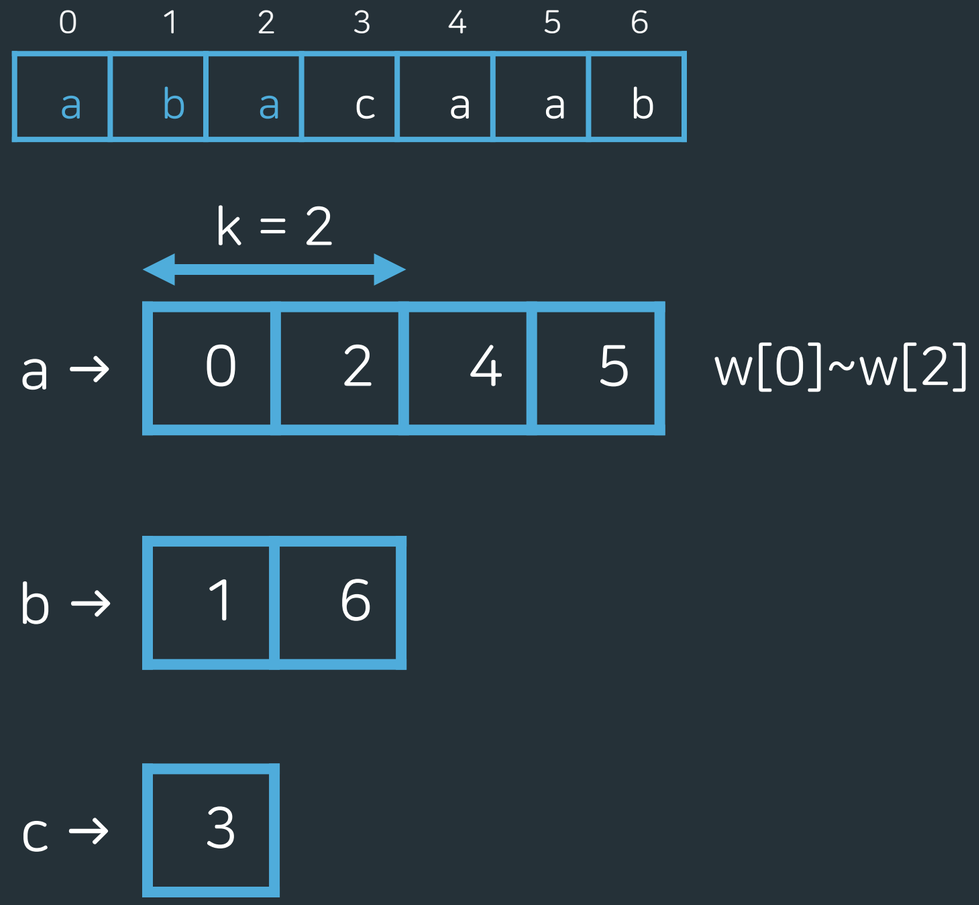
문자별로 나눈다면





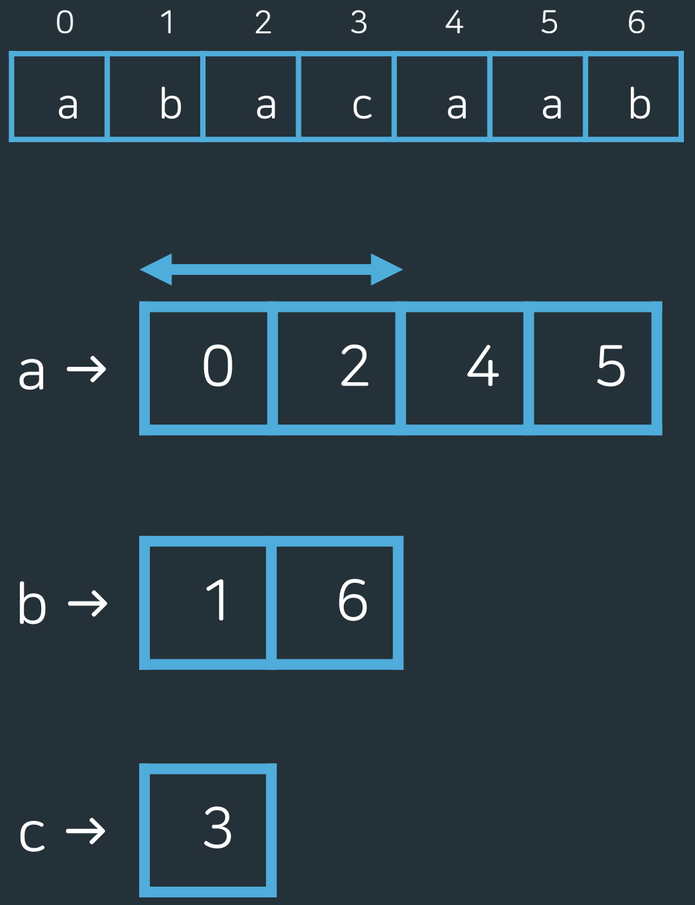
문자를 정확히 k개 포함
→ 구간의 크기가 k로 고정된 투포인터 (슬라이딩 윈도우)

문자별로 나눈다면



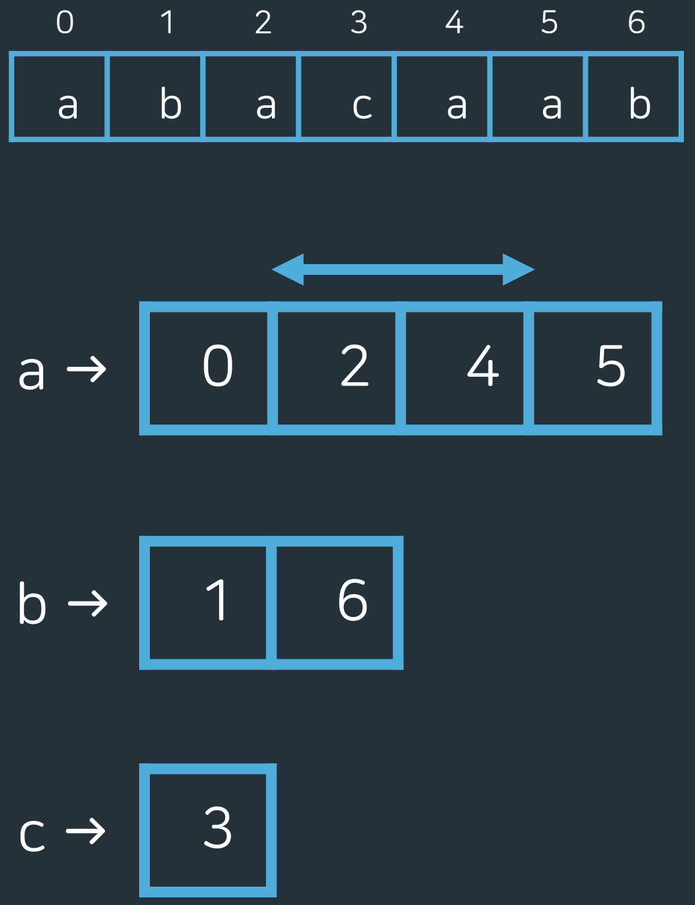
구간 내에 해당 문자를 정확히 k개 포함

슬라이딩 윈도우



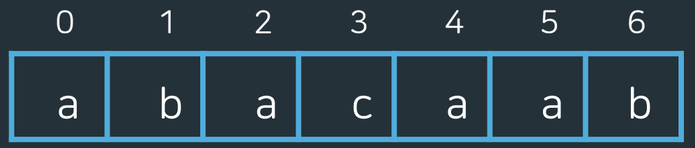
슬라이딩 윈도우를 진행하며 연속 문자열의 길이 갱신

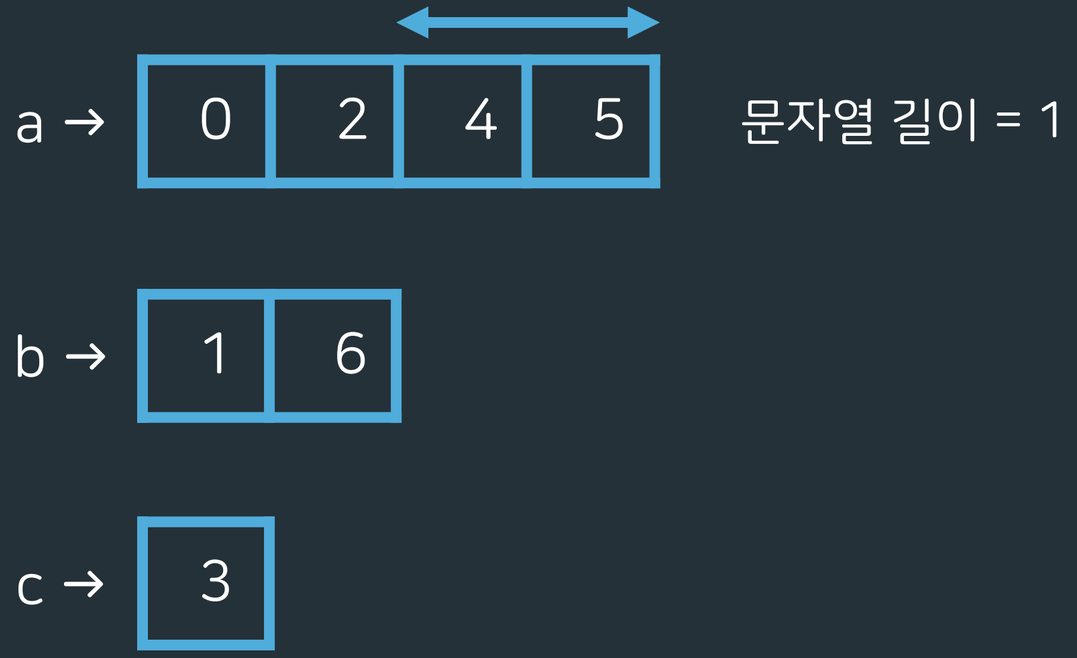
슬라이딩 윈도우



슬라이딩 윈도우를 진행하며 연속 문자열의 길이 갱신

슬라이딩 윈도우





슬라이딩 윈도우를 진행하며 연속 문자열의 길이 갱신

도전 문제 2

#알튜비튜

2473번: 세 용액 – Gold 3

문제

- 이전 수업 시간에 다루었던 두 용액 문제의 응용 버전
- 용액의 특성값의 합이 가장 0에 가까운 세 용액을 구하는 문제

제한 사항

- 전체 용액의 수 n: 3 ≤ n ≤ 1000
- 용액의 특성값: -1,000,000,000 이상 1,000,000,000 이하

예제 입력

```
5
-2 6 -97 -6 98
```

예제 출력

```
-97 -2 98
```

두 용액 문제를 떠올려 볼까요…?

#알튜비튜

2470번 : 두 용액 – Gold 5

문제

- 두 개의 서로 다른 용액을 혼합해, 합이 0에 가까운 용액을 만들어라

제한 사항

- 용액의 수 N은 2 <= N <= 100,000
- 용액의 특성값 k는 -1e9 <= k <= 1e9 (-10억 ~ 10억)

예제 입력

```
5
-2 4 -99 -1 98
```

예제 출력

```
-99 98
```

두 용액 문제를 떠올려 볼까요…?

Left + Right = 76

Ans = 76

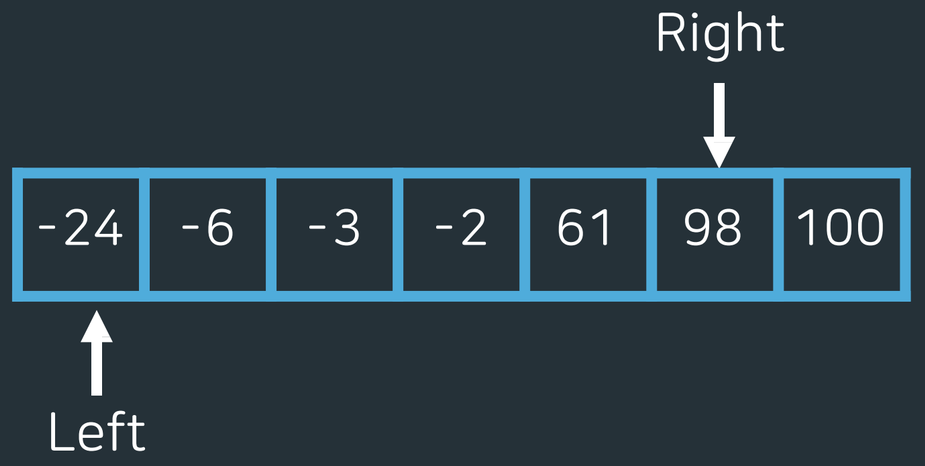
두 용액 문제를 떠올려 볼까요…?

Left + Right = 76

Ans = 76

0보다 크니까 숫자를 줄이자!

두 용액 문제를 떠올려 볼까요…?



Left + Right = 74

Ans = 74

두 용액 문제를 떠올려 볼까요…?

Left + Right = 37

Ans = 37

두 용액 문제를 떠올려 볼까요…?

Left + Right = -26

Ans = -26

두 용액 문제를 떠올려 볼까요?

#알튜비튜

- 두 용액 문제에서는 투 포인터를 두 용액으로 대응시켰었죠?

- 그렇다면 세 용액에서는…?
  ⇒ 두 용액에서와 마찬가지로 투 포인터 사용
  ⇒ 가장 주의해야 할 것은 중복이 되는 연산이 없어야 한다는 것!

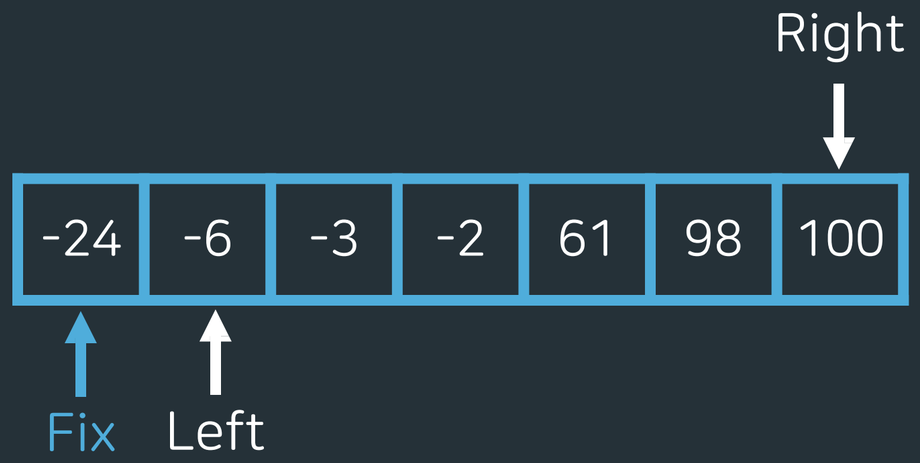
  ⇒ 반드시 포함되는 용액을 정해서 투 포인터 연산 범위를 한정시켜줍시다

먼저 모든 용액을 정렬해줍시다

투 포인터? 아니 쓰리 포인터!

투 포인터? 아니 쓰리 포인터!

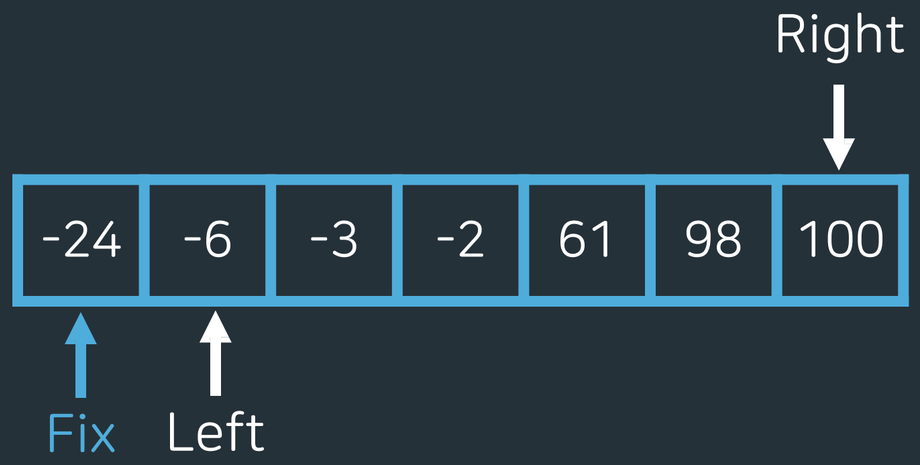
투 포인터? 아니 쓰리 포인터!



Fix 포인터로 반드시 포함할 용액 지정
Fix보다 왼쪽에 있는 용액에 대해서는 탐색 X

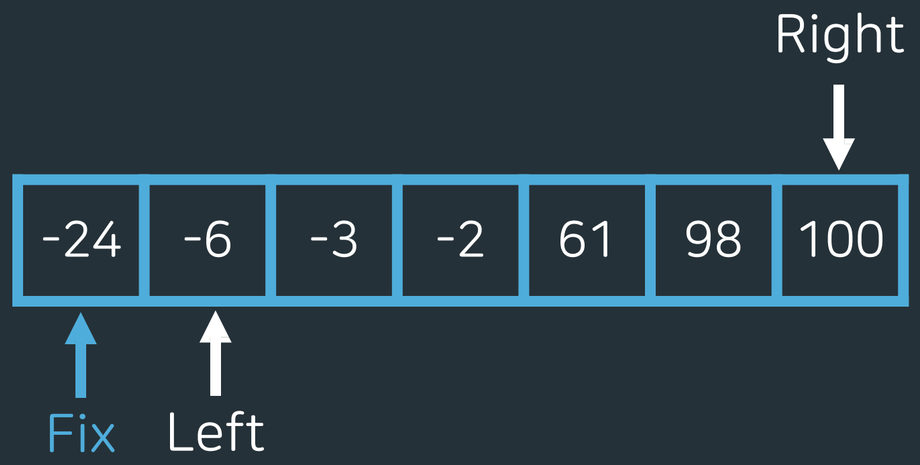
Fix를 0번째 인덱스에서부터 오른쪽으로 이동시켜가며 세 용액의 합 계산

투 포인터? 아니 쓰리 포인터!



ans = INF

투 포인터? 아니 쓰리 포인터!

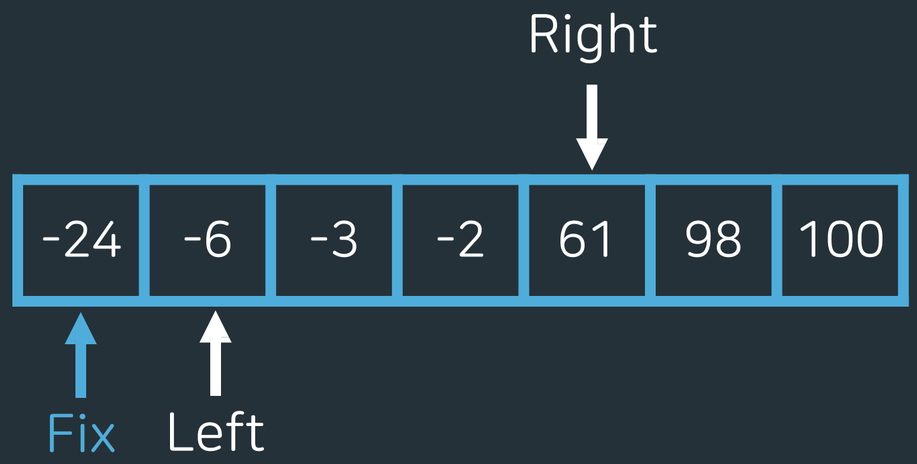


Fix + Left + Right = 70
ans = 70

투 포인터? 아니 쓰리 포인터!

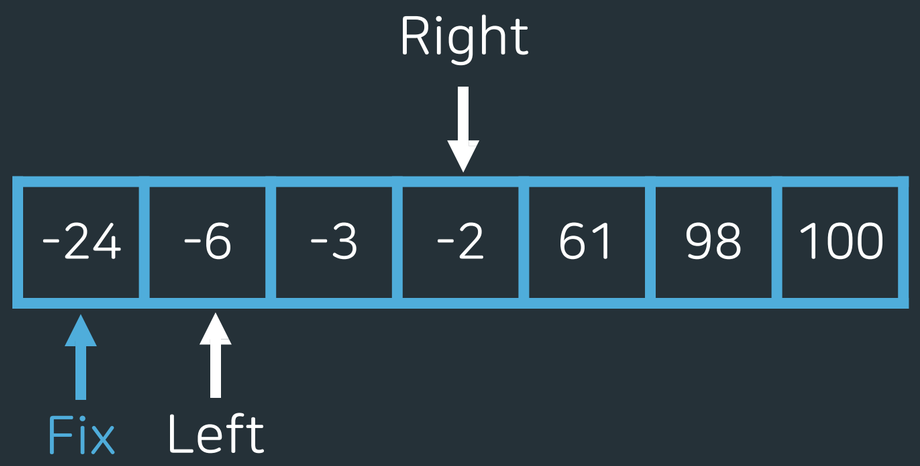
Fix + Left + Right = 68
ans = 68

투 포인터? 아니 쓰리 포인터!



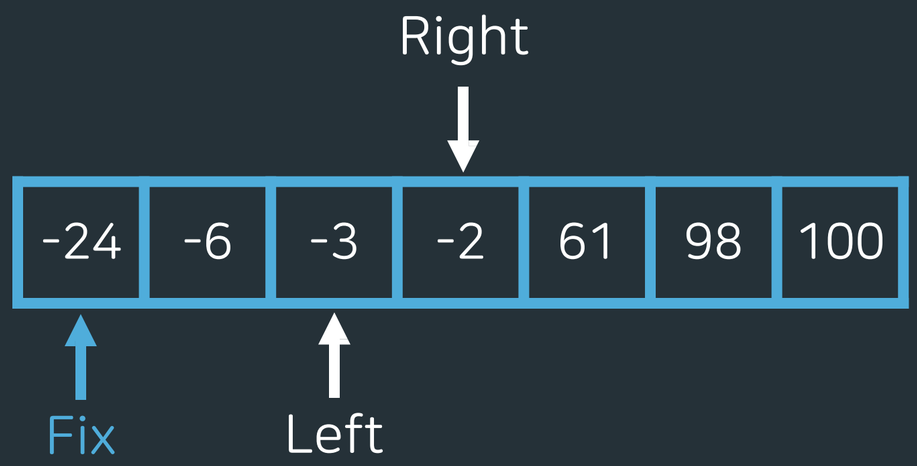
Fix + Left + Right = 31
ans = 31

투 포인터? 아니 쓰리 포인터!



Fix + Left + Right = -32
ans = 31

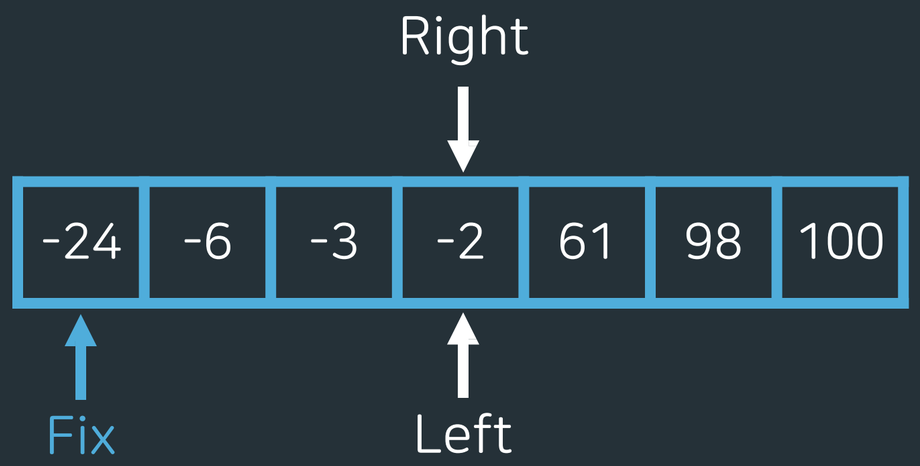
투 포인터? 아니 쓰리 포인터!



Fix + Left + Right = -29

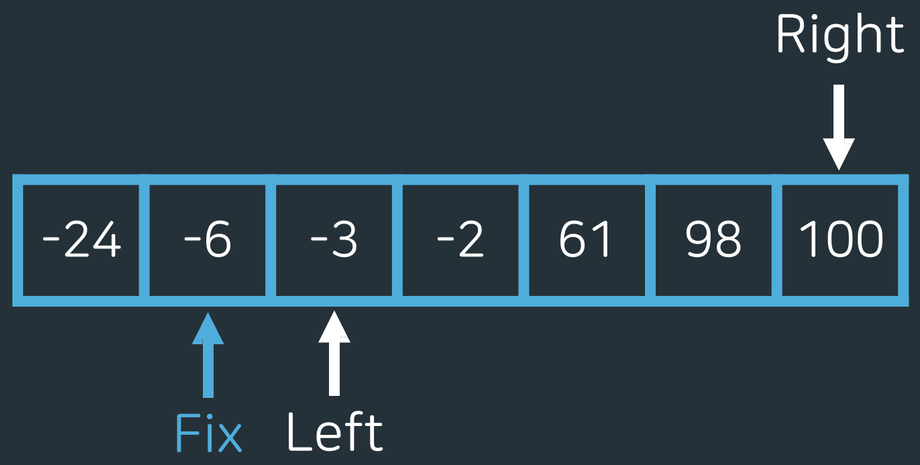
ans = -29

투 포인터? 아니 쓰리 포인터!

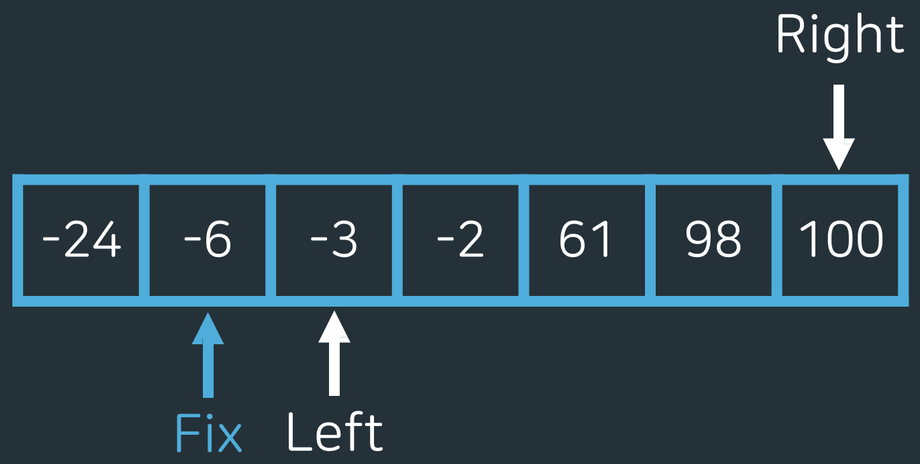


서로 다른 세 용액이어야 하므로 break
Fix를 오른쪽으로 이동

투 포인터? 아니 쓰리 포인터!



투 포인터? 아니 쓰리 포인터!



Fix + Left + Right = 91

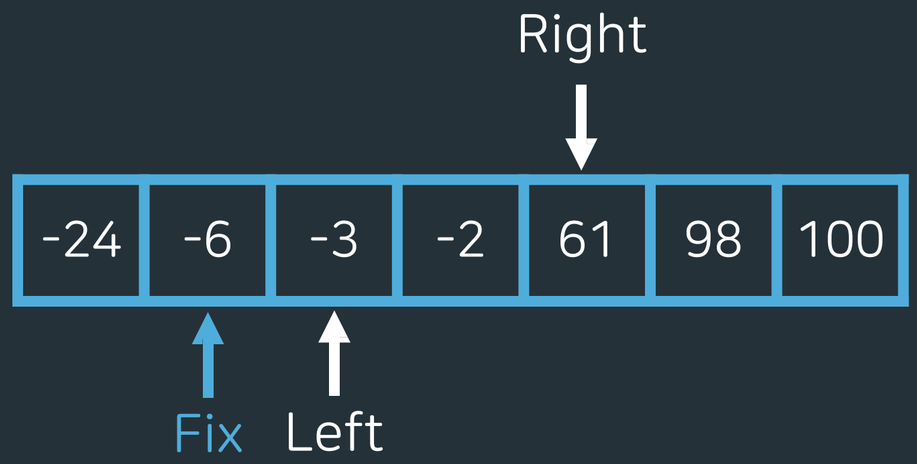
ans = -29

투 포인터? 아니 쓰리 포인터!

Fix + Left + Right = 89

ans = -29

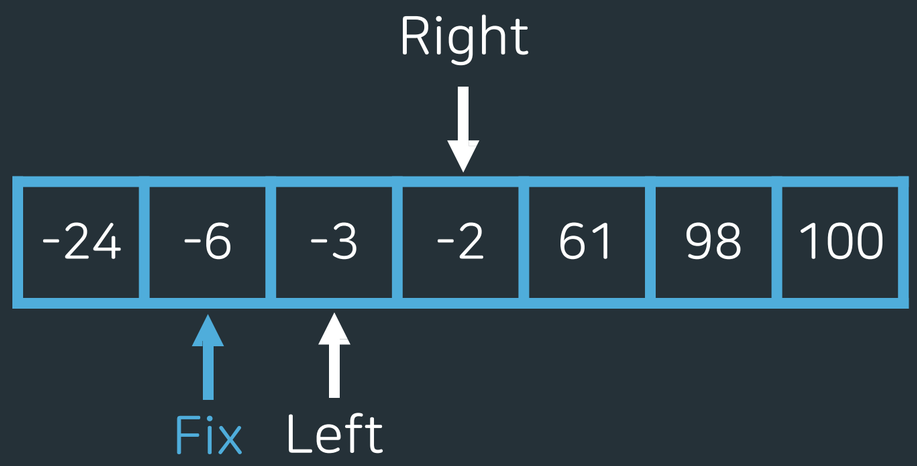
투 포인터? 아니 쓰리 포인터!



Fix + Left + Right = 52

ans = -29

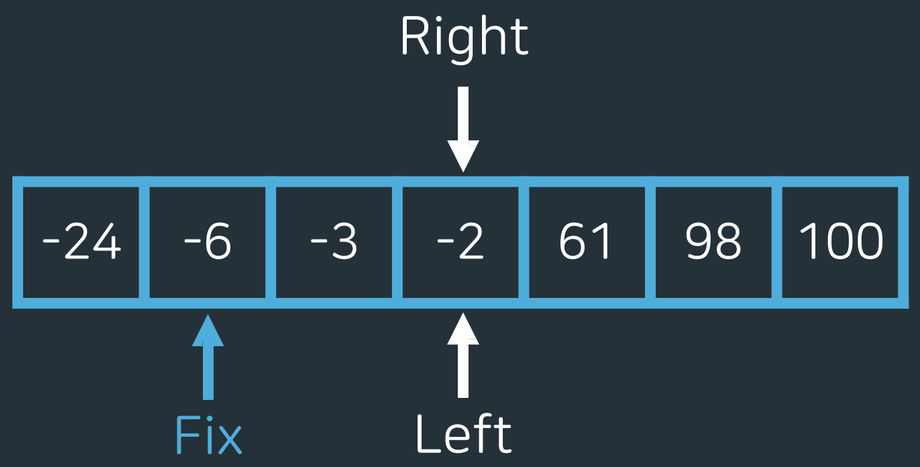
투 포인터? 아니 쓰리 포인터!



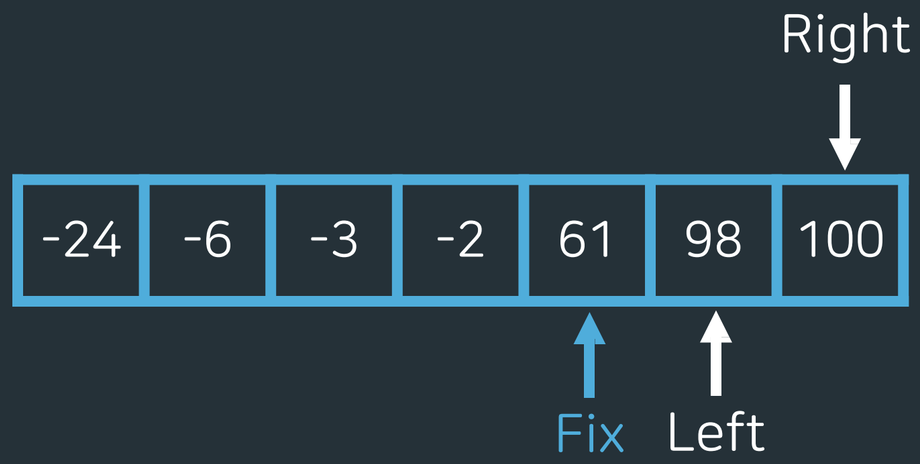
Fix + Left + Right = -11

ans = -11

투 포인터? 아니 쓰리 포인터!



투 포인터? 아니 쓰리 포인터!



Fix가 n-3이 되면 탐색 종료

/<> 14503번 : 로봇 청소기 - Gold 5

문제

로봇 청소기와 방의 상태가 주어졌을 때, 청소하는 영역의 개수를 구하는 프로그램 만들기

1. 현재 칸이 아직 청소되지 않은 경우, 현재 칸 청소
2. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우,
    1. 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 한 칸 후진하고 1번으로 돌아감
    2. 바라보는 방향의 뒤쪽 칸이 벽이라 후진할 수 없다면 작동 멈춤
3. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 있는 경우,
    1. 반시계 방향 90º 회전
    2. 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우 한 칸 전진하고 1번으로 돌아감

구현 문제

#알튜비튜

제한 사항

- 방의 크기 N과 M의 범위 3 <= N, M <= 50

예제 입력

```
3 3
1 1 0
1 1 1
1 0 1
1 1 1
```

예제 출력

```
1
```

접근법

#알튜비튜

1. 현재 칸이 아직 청소되지 않은 경우, 현재 칸 청소
2. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우,
    1. 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 한 칸 후진하고 1번으로 돌아감
    2. **바라보는 방향의 뒤쪽 칸이 벽이라 후진할 수 없다면 작동 멈춤**
3. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 있는 경우,
    1. 반시계 방향 90º 회전
    2. 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우 한 칸 전진하고 1번으로 돌아감

Hint

1. 2-2번을 만족하기 전까지 로봇 청소기는 작동을 멈추지 않아요! 무한 반복문 안에서 작동하다가 2-2번을 만족할 때 break로 반복문을 빠져나와 볼까요?
2. 문제 내용 그대로 구현하는 게 중요해요. 조건과 방향에 유의하여 구현합시다!

의사코드

1. 현재 칸이 아직 청소되지 않은 경우, 현재 칸 청소
2. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우,
    1. 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 한 칸 후진하고 1번으로 돌아감
    2. 바라보는 방향의 뒤쪽 칸이 벽이라 후진할 수 없다면 작동 멈춤
3. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 있는 경우,
    1. 반시계 방향 90º 회전
    2. 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우 한 칸 전진하고 1번으로 돌아감

```
무한반복문 {
        if(아직 청소되지 않은 경우)  // 1번 작업
                현재 칸 청소

        bool 빈칸 = false
        주변4칸 탐색 시작 { // 반시계 방향으로 회전
                if(청소되지 않은 빈칸 발견) {
                        빈칸 = true
                        한 칸 전진
                        break
                }
        }

        if(빈칸 == true) // 주변 4칸 중 빈 칸 있었음
                continue; // 1번 작업으로

        if(후진할 수 있는 경우)
                한 칸 전진

        else  // 후진할 수 없는 경우
                break // 작동 종료
}
```

의사코드

#알튜비튜

1. 현재 칸이 아직 청소되지 않은 경우, 현재 칸 청소
2. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우,
    1. 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 한 칸 후진하고 1번으로 돌아감
    2. 바라보는 방향의 뒤쪽 칸이 벽이라 후진할 수 없다면 작동 멈춤
3. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 있는 경우,
    1. 반시계 방향 90º 회전
    2. 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우 한 칸 전진하고 1번으로 돌아감

```
무한반복문 {
        if(아직 청소되지 않은 경우)  // 1번 작업
                현재 칸 청소

        bool 빈칸 = false
        주변4칸 탐색 시작 { // 반시계 방향으로 회전
                if(청소되지 않은 빈칸 발견) {
                        빈칸 = true
                        한 칸 전진
                        break
                }
        }

        if(빈칸 == true) // 주변 4칸 중 빈 칸 있었음
                continue; // 1번 작업으로

        if(후진할 수 있는 경우)
                한 칸 전진

        else  // 후진할 수 없는 경우
                break // 작동 종료
}
```

의사코드

#알튜비튜

1. 현재 칸이 아직 청소되지 않은 경우, 현재 칸 청소
2. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 없는 경우,
    1. 바라보는 방향을 유지한 채로 한 칸 후진할 수 있다면 한 칸 후진하고 1번으로 돌아감
    2. 바라보는 방향의 뒤쪽 칸이 벽이라 후진할 수 없다면 작동 멈춤
3. 현재 칸의 주변 4칸 중 청소되지 않은 빈 칸이 있는 경우,
    1. 반시계 방향 90º 회전
    2. 바라보는 방향을 기준으로 앞쪽 칸이 청소되지 않은 빈 칸인 경우 한 칸 전진하고 1번으로 돌아감

```
무한반복문 {
        if(아직 청소되지 않은 경우)  // 1번 작업
                현재 칸 청소

        bool 빈칸 = false
        주변4칸 탐색 시작 { // 반시계 방향으로 회전
                if(청소되지 않은 빈칸 발견) {
                        빈칸 = true
                        한 칸 전진
                        break
                }
        }

        if(빈칸 == true) // 주변 4칸 중 빈 칸 있었음
                continue; // 1번 작업으로

        if(후진할 수 있는 경우)
                한 칸 전진

        else  // 후진할 수 없는 경우
                break // 작동 종료
}
```

DFS로 풀이할 수는 없나요?

#알튜비튜

Hint

DFS는 최대한 깊게 탐색했다가
더 이상 탐색할 수 없는 경우, 가까운 갈림길도 돌아와 다시 탐색을 진행하는 방법이에요.
하지만 로봇 청소기는 2-2번 조건을 만족하면 즉시 작동을 멈추네요.

가까운 갈림길로 되돌아와 탐색하지 않도록 구현하면 DFS로도 풀이할 수 있어요!
DFS 풀이도 올려놓았으니 확인해보세요 ☺

마무리

#알튜비튜

추가로 풀어보면 좋은 문제!

- 6159번 : 코스튬 파티 – Silver 5
- 2531번 : 회전 초밥 – Silver 1
- 16472번 : 고냥이 – Gold 4
- 1484번 : 다이어트 – Gold 4